Ezi-SERVO® Plus-R

Closed Loop Stepping System
with Network based Motion Controller

Ezi-SERVO® Plus-R MINI

Closed Loop Stepping System

User Manual

Communication Function_Ver8

(Rev.08.05.09)

FASTECH

- Table of Contents -

1. Communication Protocols 6

1－1. Communication Functions 6

1－1－1. Communication Specifications 6

1－1－2. RS-485 Communication Protocol(Ver8) 6

1－1－3. CRC Calculation Example 7

1－1－4. Response Frame Structure and Communication Error(Ver8) 10

1－2. Structure of Frame type(Ver8) 12

1－2－1. Frame type and Data Configuration 12

1－2－2. Parameter Lists 25

1－2－3. Setup bit of Output pin 26

1－2－4. Setup bit of Input pin 27

1－2－5. Bit setup of Status Flag 28

1－2－6. Position Table Item 29

1－2－7. Information of Motor 29

1－3. Program Method 30

2. Library for PC Program(Ver8) 31

2－1. Library Configuration 31

2－2. Communication Status Window 32

2－3. Drive Link Function 37

FAS_Connect 38

FAS_Close 40

FAS_GetSlaveInfo 41

FAS_IsSlaveExist 42

2－4. Parameter Control Function 43

FAS_SaveAllParameters 44

FAS_SetParameter 46

FAS_GetParamater 47

FAS_GetROMParameter 48

2－5. Servo Control Function 49

FAS_ServoEnable 50

FAS_ServoAlarmReset 52

2－6. Control I/O Function 53

FAS_SetIOInput 54

FAS_GetIOInput 56

FAS_SetIOOutput 57

FAS_GetIOOutput 58

FAS_GetInputAssignMap 59

FAS_GetOutputAssignMap 61

FAS_ SetInputAssignMap 62

FAS_ SetOutputAssignMap 63

FAS_IOAssignMapReadROM 64

2－7. Position Control Function 65

FAS_SetCommandPos 66

FAS_SetActualPos 68

FAS_GetCommandPos 69

FAS_GetActualPos 71

FAS_GetPosError 72

FAS_GetActualVel 73

FAS_ClearPosition 74

2－8. Drive Status Control Function 75

FAS_GetIOStatus 76

FAS_GetMotionStatus 77

FAS_GetAllStatus 78

FAS_GetAxisStatus 79

FAS_GetRunPTStatus 80

2－9. Running Control Function 81

FAS_MoveStop 82

FAS_EmergencyStop 83

FAS_MoveOriginSingleAxis 84

FAS_MoveSingleAxisAbsPos 85

FAS_MoveSingleAxisIncPos 87

FAS_MoveToLimit 88

FAS_MoveVelocity 89

FAS_PositionAbsOverride 90

FAS_PositionIncOverride 92

FAS_VelocityOverride 93

FAS_AllMoveStop 94

FAS_AllEmergencyStop 95

FAS_AllMoveOriginSingleAxis 96

FAS_AllMoveSingleAxisAbsPos 97

FAS_AllMoveSingleAxisIncPos 98

FAS_MoveLinearAbsPos / FAS_MoveLinearIncPos 99

FAS_MoveSingleAxisAbsPosEx 100

FAS_MoveSingleAxisIncPosEx 102

FAS_MoveVelocityEx 103

2－10. Position Table Control Function 105

FAS_PosTableReadItem 106

FAS_PosTableWriteItem 108

FAS_PosTableWriteROM 109

FAS_PosTableReadROM 110

FAS_PosTableRunItem 111

FAS_PosTableSingleRunItem 112

FAS_PosTableReadOneItem 113

FAS_PosTableWriteOneItem.....114

3. Protocol test Program..... 115

1. Communication Protocols

1－1. Communication Functions

Ezi-SERVO Plus-R can control up to 16 axes by daisy-chain link at RS-485(two-wire).

Caution	Pay attention that when Windows goes into standby or power-save mode, serial communication is basically disconnected. When the system is recovered from standby mode, it should be connected again with serial communication. This is also applicable to the library provided.

1－1－1. Communication Specifications

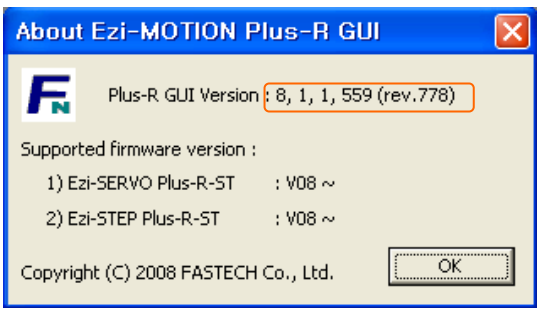
Specification	RS-485
Communication Type	Synchronous
	Half-duplex
Baudrate [bps]	9600, 19200, 38400, 57600, 115200, 230400,460800,921600
Data Type	8bit Binary
Parity	No
Stop Bit	1bit
CRC Check	Yes
Max Cabling Length (Converter ↔ Drive)	30 m
Min Cable length between drive	More than 60 cm
Number of Connected Axes	16 axes (No. 0~F)

1－1－2. RS-485 Communication Protocol(Ver8)

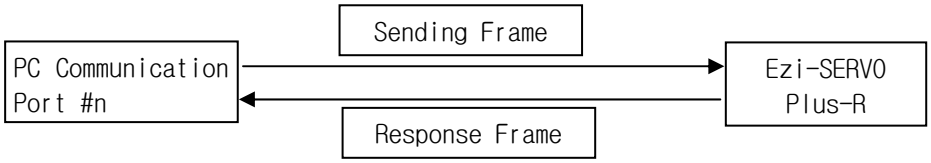
Thers are 2 kinds of program version for SERVO Plus-R. This manual support for Version 8 level.

Type	Firmware version	compatability	User Program(GUI) version
1	Level 6 (V06.0x.0xx.xx)	<->	Level 6 (6.xx.x.xxx)
2	Level 8 (V08.xx.0xx.xx)	<->	Level 8 (8.xx.x.xxx)

After connect the User Program(GUI), Version number can be check in 'About Plus-R GUI…' menu in 'Help' menu.


About Ezi-MOTION Plus-R GUI
Plus-R GUI Version : 8, 1, 1, 559 (rev.778)
Supported firmware version :
1) Ezi-SERVO Plus-R-ST : V08 ~
2) Ezi-STEP Plus-R-ST : V08 ~
Copyright (C) 2008 FASTECH Co., Ltd.
OK

1) Overview of communication FRAME



2) Basic structure of Frame

Header	*Frame Data*	Tail
0xAA 0xCC	4~252 bytes	0xAA 0xEE

① 0xAA : Delimited byte
② 0xAA 0xCC : Indicate header of the Frame.
③ 0xAA 0xEE : Indicate tail of the Frame.
④ If any of the Frame data is '0xAA', '0xAA' should be added right after it. (byte stuffing)
⑤ If any data following '0xAA' is not '0xAA', '0xCC' or '0xEE', it indicates an error.

Detailed *Frame Data* is configured as follows:

Slave ID	Sync number	Frame type	Data	CRC	
1 byte	1 byte	1 byte	0~248 bytes.	2 bytes	
				Low byte	High byte

① Slave ID : Dive module number (0~15) connected to the PC communication port.
② Sync number : Serial number for that packet.
This number have to be change for each command.
③ Frame type : Designate command type of relevant frames. For the command type, refer to 「Frame Type and Data Configuration」 section.
④ Data : Data structure and length are set according to Frame type. For more information, refer to 「Frame Type and Data Configuration」 section.
⑤ CRC : To check an error which occurs during communication, '0xA001' of a polynomial factor in CRC16(Cyclic Redundancy Check) or 'X16+X15+X2+1' of a polynomial factor in CRC-16-IBM(Cyclic Redundancy Check) is used. CRC calculation is performed for all items (Slave ID, Frame type, Data) prior to CRC item.

1-1-3. CRC Calculation Example

The following program source is included in a file (file name : CRC_Checksum.c) provided with the product.

1) '0xA001' of CRC16

```
const unsigned short TABLE_CRCVALUE[] =
{
0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
```

```
  0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
  0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
  0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
  0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
  0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
  0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
  0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
  0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
  0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
  0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
  0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
  0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
  0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
  0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
  0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
  0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
  0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
  0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
  0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
  0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
  0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
  0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
  0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
  0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
  0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040
 };

 unsigned short CalcCRC(unsigned char* pDataBuffer, unsigned long usDataLen)
{
        unsigned char nTemp;
        unsigned short wCRCWord = 0xFFFF;

        while (usDataLen--)
        {
                nTemp = wCRCWord ^ *(pDataBuffer++);
                wCRCWord >>= 8;
                wCRCWord ^= TABLE_CRCVALUE[nTemp];
        }
        return wCRCWord;
}
```

2) 'X16+X15+X2+1' of CRC-16-IBM

```
unsigned short CalcCRCbyAlgorithm(unsigned char* pDataBuffer, unsigned long usDataLen)
{
        // Use the Modbus algorithm as detailed in the Watlow comms guide
        const unsigned short POLYNOMIAL = 0xA001;
        unsigned short wCrc;
        int iByte, iBit;

        /* Initialize CRC */
        wCrc = 0xffff;
```

```
    for (iByte = 0; iByte < usDataLen; iByte++)
    {
       /* Exclusive-OR the byte with the CRC */
       wCrc ^= *(pDataBuffer + iByte);

       /* Loop through all 8 data bits */

       for (iBit = 0; iBit <= 7; iBit++)
       {
          /* If the LSB is 1, shift the CRC and XOR the polynomial mask with the CRC */

         // Note - the bit test is performed before the rotation, so can't move the << here
             if (wCrc & 0x0001)
             {
                      wCrc >>= 1;
                      wCrc ^= POLYNOMIAL;
             }
             else
             {
                      // Just rotate it
                      wCrc >>= 1;
             }
       }
    }
    return wCrc;
}
```

1－1－4. Response Frame Structure and Communication Error(Ver8)

When any command is sent, the basic structure of Frame at the response side is identical. However, there is a difference in case of *Frame Data,* which 'communication status' is added as shown below.

Slave ID	Sync number	Frame Type	Data		CRC	
1 byte	1 byte	1 byte	1 byte	0~247 bytes	2 bytes	
			Communication status	Response data	Low byte	High byte

① Slave ID : Same to sending Frame.
(When this is not same to sending data, need to recognize as the error status.)
② Sync number : Same to sending Frame
(When this is not same to sending data, need to recognize as the error status.)
③ Frame type : Same to sending Frame.
(When this is not same to sending data, need to recognize as the error status.)
④ Data : When simple executive instructions are sent, this data cannot be read. However, in case of response, 1 byte is included to the display of communication status (error / normal status).

The code by bytes means the **'Communication status'** as follows.

Hexa Code	Decimal Code	Description
0x00	0	Communication is normal status.
0x80	128	Frame Type Error : Responded Frame type cannot be recognized.
0x81	129	Data error, ROM data read/write error : Responded data value is aside from the given range.
0x82	130	Received Frame Error : Received Frame data is out of this specification.
0x85	133	Running Command Failure : The user has tried to execute new running commands in wrong condition as follows. 1) currently motor is running 2) currently motor is stopping 3) Servo is OFF status 4) try to Z-pulse Origin without external encoder
0x86	134	RESET Failure : The user has tried to execute new running commands in wrong condition as follows. 1) Servo is ON status 2) Already RESET status by external input signal
0x87	135	Servo ON Failure ① : While an alarm occurs, the user tried to execute Servo ON command.
0x88	136	Servo ON Failure ② : While Emergency Stop occurs, the user tried to execute Servo ON command.
0x89	137	Servo ON Failure ③ : 'ServoON' signal is assigned by input signal. Servo ON/OFF Can execute by external input signal only.
0xAA	170	CRC Error : When received Frame data is error by external noise, sending side of DLL Library is automatically try to send 1 more time.

Caution	1) If 'Header' and 'Slave ID' values in the sending Frame are abnormal, there is no response from the drive. 2) If the communication status is displayed to '130', the size of response data is '0' byte.

1－2. Structure of Frame type(Ver8)

1－2－1. Frame type and Data Configuration

(1) The following table explains the content and configuration by Frame type of data.

<table>
<tr><th>Frame Type</th><th>Library Name</th><th>Contents</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetSlaveInfo</td><td>
Connected slave type and program version information are required.

Sending : 0 byte
Response : 71 bytes
<table>
<tr><td>1 byte</td><td>8 bytes</td><td>30 bytes</td><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td>Communication status</td><td>Version number</td><td>Version description</td><td>Slave type</td><td>Motor type</td></tr>
</table>
<table>
<tr><td>20 byte</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td>Motor Description</td><td>Input number</td><td>Output number</td><td>PT number</td><td>Spare</td></tr>
</table>
♦ Version number and Version Description : After connect to slave with UserProgram(GUI), you can find in ‘Information’ window of ‘Board List’ window.
♦ Slave type : 1 : Ezi-SERVO Plus-R series
20 : Ezi-STEP Plus-R series
♦ Motor type : For manufacturer.
♦ Motor Description : ‘refer to 「1-2-7.Information of Motors」
♦ Input/Output number: Signal numbers that can be connect to drive.
♦ PT number : Motion step number that can be set to drive (64~256)
♦ Spare : For manufacturer.
</td></tr>
<tr><td>0x10
(16)</td><td>FAS_
SaveAllParameters</td><td>
Save currently set parameters & assigned signals in the ROM of the drive. Even the drive is powered off, saving these data & parameters are possible.
Values set at ‘FAS_SetParameter’ & ‘FAS_SetIOAssignMap’ are saved together.

Sending : 0 byte

Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td>0x11
(17)</td>
<td>FAS_
GetROMParameter</td>
<td>Specific parameter value in the ROM are recognized.

Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~31)</td></tr></table>
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table>
Refer to 「1-2-2.Parameter List」</td>
</tr>
<tr>
<td>0x12
(18)</td>
<td>FAS_
SetParameter</td>
<td>Specific parameter value are saved to the RAM.

Sending : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Parameter number (0~31)</td><td>Parameter value</td></tr></table>
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>
Refer to 「1-2-2.Parameter List」</td>
</tr>
<tr>
<td>0x13
(19)</td>
<td>FAS_
GetParameter</td>
<td>Specific parameter value in the RAM are recognized.

Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~31)</td></tr></table>
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table>
Refer to 「1-2-2.Parameter List」</td>
</tr>
</table>

<table>
<tr>
<td>0x20
(32)</td>
<td>FAS_
SetIOOutput</td>
<td>Set output signal level of the control output port.

Sending : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr>
</table>
When specific bit of the ‘set mask’ is ‘1’, the relevant output port signal is set to [ON].
When specific bit of the ‘clear mask’ is ‘1’, the relevant output port signal is set to [OFF].
For more information, refer to 「1-2-3.Bit setup of Output Pin」.

Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x21
(33)</td>
<td>FAS_
SetIOInput</td>
<td>Set input signal level of the control input port.

Sending : 10 bytes
<table>
<tr><td>5 bytes</td><td>5 bytes</td></tr>
<tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr>
</table>
When specific bit of the ‘set mask’ is ‘1’, the relevant input port signal is set to [ON].
When specific bit of the ‘clear mask’ is ‘1’, the relevant input port signal is set to [OFF].
For more information, refer to 「1-2-4. Bit setup of Input Pin」.

Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x22
(34)</td>
<td>FAS_
GetIOInput</td>
<td>Current input signal status of the control input port is recognized.

Sending : 0 byte
Response : 6 byte
<table>
<tr><td>1 byte</td><td>5 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td></tr>
</table>
Relevant bit by each input signal, refer to 「1-2-4. Bit setup of Input Pin」.</td>
</tr>
<tr>
<td>0x23
(35)</td>
<td>FAS_
GetIOOutput</td>
<td>Current output signal status of the control output port is recognized.

Sending : 0 byte
Response : 5 byte
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Output status value</td></tr>
</table>
Relevant bit by each output signal, refer to 「1-2-3.Bit setup of Output Pin」.</td>
</tr>
</table>

<table>
<tr><td>0x24
(36)</td><td>FAS_
SetInputAssignMap</td><td>Assign Input signal to the pin of CN1 port and set signal level simultaneously. By running ‘FAS_SaveAllParameters’, you can save the setting value to the ROM.

Sending : 3 bytes
<table><tr><td>1 byte</td><td>1 bytes</td><td>1 byte</td></tr><tr><td>IN number</td><td>Assign number</td><td>Setting level</td></tr></table>
♦ IN number: ‘0~11’ corresponds to ‘Limit+, Limit-, Org,IN1,…, IN9’ respectively.
♦ Assign number: 1~35 (refer to 「1-2-4. Setup bit of Input」)
♦ Level Setting: 0:Active Low, 1:Active High

Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x25
(37)</td><td>FAS_
SetOutputAssignMap</td><td>Assign Output signal to the pin of CN1 port and set signal level simultaneously. By running ‘FAS_SaveAllParameters’, you can save the setting value to the ROM.

Sending : 3 bytes
<table><tr><td>1 byte</td><td>1 bytes</td><td>1 byte</td></tr><tr><td>OUT number</td><td>Assign number</td><td>Setting level</td></tr></table>
♦ OUT number: ‘0~9’ corresponds to ‘COMP, OUT1,…, OUT9’ respectively.
♦ Assign number: 1~24 (refer to 「1-2-3. Setup bit of Output」)
♦ Level Setting: 0:Active Low, 1:Active High

Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x26
(38)</td><td>FAS_
IOAssignMapReadROM</td><td>Recognize setting status of I/O & setting value from ROM area. These values are loaded to RAM.

Sending : 0 byte

Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x2A
(42)</td><td>FAS_
ServoEnable</td><td>Set ON/OFF status.

Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>0:OFF, 1:ON</td></tr></table>
♦ After ‘ServoON’ is executed, the ‘Command Position’ value will change as same as ‘Actual Position’ value to remove ‘Position Error’.

<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table> Response : 1 byte</td></tr>
</table>

<table>
<tr><td>0x2B
(43)</td><td>FAS_
ServoAlarmReset</td><td>Reset Servo alarm status.

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x2E
(46)</td><td>FAS_
ServoAlarmtype</td><td>To request the Alarm type.
Sending: 0 byte
Response: 2 byte<table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>Communication status</td><td>Alarm type (0~)</td></tr></table>
♦ Alarm type: No alarm (0) OverCurrent(1) OverSpeed(2)
StepOut(3) OverLoad(4) OverTemperature(5)
BackEMF(6) MotorConnect(7) EncoderConnect(8)
MotorPower(9) Inposition(10) SystemHalt(11)
ROMdevice(12) OverInputVoltage(14)</td></tr>
<tr><td>0x31
(49)</td><td>FAS_
MoveStop</td><td>Request to stop motor currently operates.

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x32
(50)</td><td>FAS_
EmergencyStop</td><td>Request Emergency stop of the motor.

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x33
(51)</td><td>FAS_
MoveOriginSingleAxis</td><td>Request the motor to return to origin under current setting parameter condition

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x34
(52)</td><td>FAS_
MoveSingleAxisAbsPos</td><td>Request the motor to move its position as much as the absolute value[pulse]

Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x35
(53)</td><td>FAS_
MoveSingleAxisIncPos</td><td>Request the motor to move its position as much as the incremental value[pulse]
Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Incremental position value</td><td>Running speed [pps]</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x36
(54)</td><td>FAS_
MoveToLimit</td><td>Request the motor to start limit motion under current setting parameter condition

Sending : 5 bytes<table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed [pps]</td><td>Running direction (0: -Limit 1: +Limit)</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x37
(55)</td><td>FAS_
MoveVelocity</td><td>Request the motor to start jog motion at the current setting parameter condition

Sending : 5 bytes<table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed [pps]</td><td>Running direction (0: -Jog 1: +Jog)</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x38
(56)</td><td>FAS_
PositionAbsOverride</td><td>Request the motor to change the target absolute position value[pulse] while it is in running.

Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x39
(57)</td><td>FAS_
PositionIncOverride</td><td>Request the motor to change the target incremental position value[pulse] during operation.

Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x3A
(58)</td><td>FAS_
VelocityOverride</td><td>Request the motor to change the running speed value[pps] while it is in running.
Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed running speed [pps]</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x3B
(59)</td><td>FAS_
AllMoveStop</td><td>Request stop for all motor that connected in same port during operation.

Sending : 0 byte
(Slave number must be ‘99’)
Response : no response
(All slaves does not send response because can not receive response from all slaves simultaneously)</td></tr>
<tr><td>0x3C
(60)</td><td>FAS_
AllEmergencyStop</td><td>Request emergency stop for all motor that connected in same port during operation.

Sending : 0 byte
(Slave number must be ‘99’)
Response : no response
(All slaves does not send response because can not receive response from all slaves simultaneously)</td></tr>
<tr><td>0x3D
(61)</td><td>FAS_All
MoveOriginSingleAxis</td><td>Request return to origin under current setting parameter condition for all drives that connected in same port.

Sending : 0 byte
(Slave number must be ‘99’)
Response : no response
(All slaves does not send response because can not receive response from all slaves simultaneously)</td></tr>
<tr><td>0x3E
(62)</td><td>FAS_All
SingleAxisAbsPos</td><td>Request move its position as much as the absolute value[pulse] for all drives that connected in same port.

Sending : 8 bytes (Slave number must be ‘99’)<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td></tr></table>Response : no response
(All slaves does not send response because can not receive response from all slaves simultaneously)</td></tr>
</table>

<table>
<tr>
<td>0x3F
(63)</td>
<td>FAS_All
SingleAxisIncPos</td>
<td>Request move its position as much as the incremental value[pulse] for all motors that connected in same port.

Sending : 8 bytes (Slave number must be ‘99’)
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>incremental position value</td><td>Running speed [pps]</td></tr>
</table>

Response : no response
(All slaves does not send response because can not receive response from all slaves simultaneously)</td>
</tr>
<tr>
<td>0x80
(128)</td>
<td>FAS_
MoveSingleAxisAbsPos
Ex</td>
<td>Request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec]

Sending: 40 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr>
<tr><td>Absolute position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr>
</table>
<table>
<tr><td>2 bytes</td><td>24 bytes</td></tr>
<tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>

Flag ooption : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.

Response: 1 byte</td>
</tr>
</table>

<table>
<tr>
<td>0x81
(129)</td>
<td>FAS_
MoveSingleAxisIncPos
Ex</td>
<td>Request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec]

Sending: 40 bytes

<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr>
<tr><td>incremental position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr>
</table>
<table>
<tr><td>2 bytes</td><td>24 bytes</td></tr>
<tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>
Flag ooption : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.

Response: 1 byte</td>
</tr>
<tr>
<td>0x82
(130)</td>
<td>FAS_
MoveVelocityEx</td>
<td>Request the motor to start jog motion at the current setting parameter condition with custom Accel/Decel time value[msec].

Sending: 37 bytes

<table>
<tr><td>4 bytes</td><td>1 bytes</td><td>4 bytes</td></tr>
<tr><td>Running speed [pps]</td><td>Running direction (0: -Jog 1: +Jog)</td><td>Flag option</td></tr>
</table>
<table>
<tr><td>2 bytes</td><td>26 bytes</td></tr>
<tr><td>Custom Accel./Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>
Flag ooption : 0x0001 : reserved
0x0002 : Custom Accel./Decel. Time is used.

If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.

Response : 1 byte</td>
</tr>
</table>

<table>
<tr>
<td>0x40
(64)</td>
<td>FAS_
GetAxisStatus</td>
<td>Request the Flag value indicates operation status.

Sending : 0 byte
Response : 5 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Status flag value</td></tr>
</table>

Assigned bit related to each Flag, refer to 「1-2-5. Bit setup of Status Flag」.</td>
</tr>
<tr>
<td>0x41
(65)</td>
<td>FAS_
GetIOStatus</td>
<td>Request the I/O status and the running Flag status.
(Frame type 0x22 and 0x23 are packed.)

Sending : 0 byte
Response : 10 bytes
<table>
<tr><td>1 byte</td><td>5 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td><td>Output status value</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x42
(66)</td>
<td>FAS_
GetMotionStatus</td>
<td>Request the current operation progress status and its Position Table number.
(Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)

Sending : 0 byte
Response : 21 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Command position value</td><td>Actual Position value</td><td>Position Difference value</td><td>Running speed value</td><td>Current running PT number</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x43
(67)</td>
<td>FAS_
GetAllStatus</td>
<td>Request all data including the current running status
(Frame type 0x41, 0x42 and 0x44 are packed.)

Sending : 0 byte
Response : 34 bytes
<table>
<tr><td>1 byte</td><td>5 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td><td>Output status value</td><td>Status flag value</td></tr>
</table>

<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Command position value</td><td>Actual position value</td><td>Position Difference value</td><td>Running speed value</td><td>Current running PT number</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td>0x44
(68)</td><td>FAS_
GetPTStatus</td><td>Request the current running PT number.

Sending : 0 byte
Response: 5 byte
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Current running PT number</td></tr></table>
♦ PT number : 0~255 (if there's no running : '65535' (0xFFFF))</td></tr>
<tr><td>0x50
(80)</td><td>FAS_
SetCommandPos</td><td>Ezi-SERVO Plus-R is the closed loop control system so the command position value is continuously controlled while motor is operating. User can set the command position value before it starts and then can check how the command position value is changed.

Sending : 4 bytes
<table><tr><td>4 bytes</td></tr><tr><td>Command position setting count value</td></tr></table>Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x51
(81)</td><td>FAS_
GetCommandPos</td><td>Request the command position value[pulse] being tracked.

Sending : 0 byte
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Command position value</td></tr></table></td></tr>
<tr><td>0x52
(82)</td><td>FAS_
SetActualPos</td><td>Ezi-SERVO Plus-R is the closed loop control system so the actual position value is continuously renewed while the motor is operating. User can set the actual position value before it starts and then can check how the actual position value is changed.

Sending : 4 bytes
<table><tr><td>4 bytes</td></tr><tr><td>Actual position count value</td></tr></table>Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x53
(83)</td><td>FAS_
GetActualPos</td><td>Request the current actual position value[pulse].

Sending : 0 byte
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Actual position value</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x54
(84)</td><td>FAS_
GetPosError</td><td>Request the difference[pulse] between the command position value and the actual position value.

Sending : 0 byte
Response : 5 bytes

<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Position difference value</td></tr></table>
With this value, user can check the current running status (In-position tracking status).</td></tr>
<tr><td>0x55
(85)</td><td>FAS_
GetActualVel</td><td>Request the current running speed value [pps]

Sending : 0 byte
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Speed value</td></tr></table></td></tr>
<tr><td>0x56
(86)</td><td>FAS_
ClearPosition</td><td>Ezi-SERVO Plus-R is the closed loop control system so the command position value is continuously renewed during operation. User can set the command position and actual position value as ‘0’ before it starts to operate and can check how the command position value is changed.

Sending : 0 byte
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x60
(96)</td><td>FAS_
PosTableReadItem</td><td>Read Position Table values in the RAM of the drive.

Sending : 2 bytes
<table><tr><td>2 bytes</td></tr><tr><td>Readable PT number (0~255)</td></tr></table>Response : 65 bytes
<table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>Communication status</td><td>Relevant PT values</td></tr></table>For items by each PT, refer to 「1-2-6. Position Table Item」 .</td></tr>
<tr><td>0x61
(97)</td><td>FAS_
PosTableWriteItem</td><td>To save Position Table values to the RAM of the drive.

Sending : 66 bytes
<table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT number (0~255)</td><td>Relevant PT value</td></tr></table>For items by each PT, refer to 「1-2-6. Position Table Item」 .
Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x62
(98)</td><td>FAS_
PosTableReadROM</td><td>Read all Position Table values (256 ea) in the ROM of the drive

Sending : 0 byte
Response : 2 bytes<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x63
(99)</td><td>FAS_
PosTableWriteROM</td><td>Save all Position Table value(256 ea) to the ROM of the drive.

Sending : 0 byte
Response : 2 bytes<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x64
(100)</td><td>FAS_
PosTableRunItem</td><td>Start the position table operation from the designated Position Table number

Sending : 2 bytes<table><tr><td>2 bytes</td></tr><tr><td>PT Number (0~255)</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x68
(104)</td><td>FAS_
PosTableSingleRunItem</td><td>Start the specific only one position table operation that is designated Position Table number
Sending : 3 bytes<table><tr><td>1 byte</td><td>2 bytes</td></tr><tr><td>Select Jump function (0:No Jump,1:Jump)</td><td>PT Number (0~255)</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x6A
(106)</td><td>FAS_
PosTableReadOneItem</td><td>Read one of Position Table values in the RAM of the drive.
Sending: 4 byte<table><tr><td>2 byte</td><td>2 byte</td></tr><tr><td>PT Number (0~255)</td><td>Offset value(0~40)</td></tr></table>Refer to 「1-2-6. Position Table Item」 for Offset value
Response: 5 byte<table><tr><td>1 byte</td><td>4 byte</td></tr><tr><td>Communication status</td><td>Relevant one of PT value</td></tr></table></td></tr>
<tr><td>0x6B
(107)</td><td>FAS_
PosTableWriteOneItem</td><td>Save one of Position Table values to the RAM of the drive.
Sending: 8 byte<table><tr><td>2 byte</td><td>2 byte</td><td>4 byte</td></tr><tr><td>PT Number (0~255)</td><td>Offset value (0~40)</td><td>Relevant one of PT value</td></tr></table>Refer to 「1-2-6. Position Table Item」 for Offset value
Response: 2 byte<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr></table></td></tr>
</table>

* Frame Type ‘0x65’ ~ ‘0x69’, ‘0x90’ ~ ‘0x92’ are assigned for internal use.

1－2－2. Parameter Lists

No.	Name	Unit	Lower Limit	Upper Limit	Default
0	Pulse per Revolution		0	9	9
1	Axis Max Speed	[pps]	1	2,500,000	500,000
2	Axis Start Speed	[pps]	1	35,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9999	100
5	Speed Override	[%]	1	500	100
6	Jog Speed	[pps]	1	500,000	5,000
7	Jog Start Speed	[pps]	1	35,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	Servo Alarm Logic		0	1	0
10	Servo ON Logic		0	1	0
11	Servo Alarm Reset Logic		0	1	0
12	S/W Limit Plus Value	[pulse]	-134,217,727	+134,217,727	+134,217,727
13	S/W Limit Minus Value	[pulse]	-134,217,727	+134,217,727	-134,217,727
14	S/W Limit Stop Method		0	1	1
15	H/W Limit Stop Method		0	1	1
16	Limit Sensor Logic		0	1	0
17	Org Speed	[pps]	1	500,000	5,000
18	Org Search Speed	[pps]	1	500,000	1,000
19	Org Acc Dec Time	[msec]	1	9,999	50
20	Org Method		0	2	0
21	Org Dir		0	1	0
22	Org Offset	[pulse]	-134,217,727	+134,217,727	0
23	Org Position Set	[pulse]	-134,217,727	+134,217,727	0
24	Org Sensor Logic		0	1	0
25	Position Loop Gain		0	15	4
26	Inpos Value		0	15	0
27	Pos Tracking Limit	[pulse]	1	+134,217,727	5,000
28	Motion Dir		0	1	0
29	Limit Sensor Dir		0	1	0
30	Org Torque Ratio	[%]	10	100	50
31	Motor Lead		1	50	1
32	Gear Ratio		1	50	1
33	Pos. Error Overflow Limit	[pulse]	1	+134,217,727	5,000

1－2－3. Setup bit of Output pin

Here is detail description of '0x20', '0x23', '0x25', '0x28' Frame type.
This command is only applicable to 9 signals of 'User Output 0' ~ ' User Output 8' out of 24 signal types in the control output port. The rest of 15 output signal cannot be operated by the user's disposal. When any relevant situation occurs while the drive operates, they will be indicated. The following table shows bit mask values by each signal.

Signal Name	Assign number	Bit Position	Signal Name	Assign number	Bit Position	Signal Name	Assign number	Bit Position
Compare Out	1	0x00000001	Origin Search OK	9	0x00000100	User Output 1	17	0x00010000
Inposition	2	0x00000002	ServoReady	10	0x00000200	User Output 2	18	0x00020000
Alarm	3	0x00000004	reserved	11	0x00000400	User Output 3	19	0x00040000
Moving	4	0x00000008	Brake	12	0x00000800	User Output 4	20	0x00080000
Acc/Dec	5	0x00000010	PT Output0	13	0x00001000	User Output 5	21	0x00100000
ACK	6	0x00000020	PT Output1	14	0x00002000	User Output 6	22	0x00200000
END	7	0x00000040	PT Output2	15	0x00004000	User Output 7	23	0x00400000
AlarmBlink	8	0x00000080	User Output 0	16	0x00008000	User Output 8	24	0x00800000

【Example 1】 Sending data to turn ON the User Output 5.

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00100000	0x00000000

【Example 2】 Sending data to turn OFF the User Output 5

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00100000

1－2－4. Setup bit of Input pin

Here is detail description of 0x21', '0x22', '0x24', '0x27' Frame type.
This command is only applicable to 35 signals in the control input port. User can use signals for testing as if they are inputted without actual input signal. The following table shows bit mask values by each signal.

Signal Name	Assign number	Bit Position	Assign number	신호 할당 값	Bit Position	Signal Name	Assign number	Bit Position
Limit+	1	0x00000001	PT A4	9	0x00000100	AlarmReset	17	0x00010000
Limit-	2	0x00000002	PT A5	10	0x00000200	ServoON	18	0x00020000
Origin	3	0x00000004	PT A6	11	0x00000400	Pause	19	0x00040000
Clear Position	4	0x00000008	PT A7	12	0x00000800	Org Search	20	0x00080000
PT A0	5	0x00000010	PT Start	13	0x00001000	Teaching	21	0x00100000
PT A1	6	0x00000020	Stop	14	0x00002000	E-stop	22	0x00200000
PT A2	7	0x00000040	Jog+	15	0x00004000	JPT input0	23	0x00400000
PT A3	8	0x00000080	Jog-	16	0x00008000	JPT input1	24	0x00800000

Signal Name	Assign number	Bit Position
JPT input2	25	0x01000000
JPT Start	26	0x02000000
User Input0	27	0x04000000
User Input1	28	0x08000000
User Input2	29	0x10000000
User Input3	30	0x20000000
User Input4	31	0x40000000
User Input5	32	0x80000000
User Input6	33	0x100000000
User Input7	34	0x200000000
User Input8	35	0x400000000

【Example 1】 Sending data to turn ON the Pause port

5 bytes (I/O set mask value)	5 bytes (I/O clear mask value)
0x000000040000	0x000000000000

【Example 2】 Sending data to turn OFF the Pause port

5 bytes (I/O set mask value)	5 bytes (I/O clear mask value)
0x000000000000	0x000000040000

1-2-5. Bit setup of Status Flag

Refer to 'motion_define.h' of include files.

Name of Flag Define	Contents	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0x00000001
FFLAG_HWPOSILMT	'+' direction limit sensor turns ON.	0x00000002
FFLAG_HWNEGALMT	'-' direction limit sensor turns ON.	0x00000004
FFLAG_SWPOGILMT	'+' direction program limit is exceeded.	0x00000008
FFLAG_SWNEGALMT	'-' direction program limit is exceeded.	0x00000010
reserved		0x00000020
FFLAG_ERROVERCURRENT	The motor driving device is under over-current.	0x00000040
FFLAG_ERROVERSPEED	The motor speed exceeded 3000[rpm].	0x00000080
FFLAG_ERRPOSTRACKING	The motor is not tracked normally by pulse input.	0x00000100
FFLAG_ERROVERLOAD	Load exceeding the max torque of the motor is loaded more than 5 seconds.	0x00000200
FFLAG_ERROVERHEAT	The internal temperature of the drive exceeds 55° C.	0x00000400
FFLAG_ERRBACKEMF	A counter electromotive force of the motor exceeds 70V.	0x00000800
FFLAG_ERRMOTORCONNECT	Power is ON without connection of motor cable to drive	0x00001000
FFLAG_ERRENCCONNECT	Cable connection error with Encoder connector in drive	0x00002000
FFLAG_ERRMOTORPOWER	The power supplied to the motor is less than low limited value.	0x00004000
FFLAG_ERRINPOSITION	After operation is finished, a position error occurs for more than 2 seconds.	0x00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0x00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0x00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0x00040000
FFLAG_INPOSITION	Inposition has been finished.	0x00080000
FFLAG_SERVOON	The motor is under Servo ON.	0x00100000
FFLAG_ALARMRESET	AlarmReset has run.	0x00200000
FFLAG_PTSTOPED	Position Table operation has been finished.	0x00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0x00800000
FFLAG_ZPULSE	The motor operates to z-pulse type of origin return operations.	0x01000000
FFLAG_ORIGINRETOK	Origin return operation has been finished.	0x02000000
FFLAG_MOTIONDIR	To display the motor operating direction (+: OFF, -: ON)	0x04000000
FFLAG_MOTIONING	The motor is running.	0x08000000
FFLAG_MOTIONPAUSE	The motor in running is stopped by Pause command.	0x10000000
FFLAG_MOTIONACCEL	The motor is operating to the acceleration section.	0x20000000
FFLAG_MOTIONDECEL	The motor is operating to the deceleration section.	0x40000000
FFLAG_MOTIONCONST	The motor is not running as acceleration/deceleration but as constant speed of operation.	0x80000000

1-2-6. Position Table Item

Refer to 'motion_define.h' of include files.

Name	Name of Structure Parameter	Number of Bytes	Offset position	Unit	Low Limit	Upper Limit
Position	lPosition	4 (signed)	0	[pulse]	-134217728	+134217728
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	500000
High Speed	dwMoveSpd	4 (unsigned)	8	[pps]	0	500000
Accel. Time	wAccelRate	2 (unsigned)	12	[msec]	1	9999
Decel. Time	wDecelRate	2 (unsigned)	14	[msec]	1	9999
Command	wCommand	2 (unsigned)	16		0	9
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	600000
Continuous Action	wContinuous	2 (unsigned)	20		0	1
Jump Table No.	wBranch	2 (unsigned)	22		0 10000	255 10255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10000	255 10255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10000	255 10255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10000	255 10255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10000	255 10255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Check Inposition	bCheckInpos	2 (unsigned)	38		0	1
Blank		24 (unsigned)	40	0x00		

Please refer to separate manual 「User Manual_Position Table」 for setting method per each item.

1-2-7. Information of Motor

First 2 digit of number and 1~2 characters indicate the motor size and length..

【Example】 56XL : Motor Flange size is 56mm and Extra long size

Another part indicates the motor manufacturer information as below.

Display	Manufacture	Display	Manufacture
JS	JapanServo	SD	Sanyo Denki
POR	Portescap	NPM	NPM
FUL	Fulling	DI	DINGS

1－3. Program Method

There are 2 method of programming for Ezi-SERVO Plus-R.

The first is generally used method with using Visual C++ language under window system of PC. Library that serviced together with Ezi-SERVO Plus-R have to be used. Please refer to 「2. Library for PC Program」

The second method is sending command(characters) directly to Ezi-SERVO Plus-R. The user has to prepare low level protocol programming like 'Protocol Test' program and this method is applied when use higher level control unit as like PLC.
For more programming method details, please exercise 'Protocol Test_PlusR.exe' is serviced together with GUI.
Please refer to 「3. Protocol for PLC Program」.

2. Library for PC Program(Ver8)

2－1. Library Configuration

To use this library, C++ header file(*.h) and library file(*.lib or *.dll) are required. These files locate in "WWFASTECHWWEziMOTION PlusR V8WWincludeWW And the following contents should be included in a source file for development.

#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWFAS_EziMotionPlusR.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWCOMM_Define.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWMOTION_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWReturnCodes_Define.h"

And include one of next files according to drive type.

#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWMOTION_EziSERVO_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWMOTION_EziSTEP_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWMOTION_EziSERVO_ABS_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusR V8WWincludeWWMOTION_EziMotionLink_DEFINE.h"

Also, library files are as follows:

"WWFASTECHWW EziMOTION PlusR V8WWincludeWWEziMotionPlusR.lib"
"WWFASTECHWW EziMOTION PlusR V8WWincludeWWEziMotionPlusR.dll"

A sample program source with using these libraries locate at
"WWFASTECHWWEziMOTION PlusR V8WWExamplesWW" folder.

(1) The following table explains values returned when each library(DLL) function is used. The user can only check the values returned at the library(DLL) function. Low level programming Method does not support following table.

Item	Definition	Returned	Description
Normal	FMM_OK	0	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	1	Wrong port number is inputted.
	FMM_INVALID_PORT_NUM	2	The port that is not connected.
	FMM_INVALID_SLAVE_NUM	3	Wrong slave number is inputted.
Operation Error	FMM_POSTABLE_ERROR	9	An error occurs while the motor accesses to the position table.
Connection Error	FMC_DISCONNECTED	5	The relevant drive is disconnected.
	FMC_TIMEOUT_ERROR	6	Response delay(100 msec) occurs.
	FMC_CRCFAILED_ERROR	7	Checksum error occurs.
	FMC_RECVPACKET_ERROR	8	Protocol level error occurs in packet that comes from Drive.

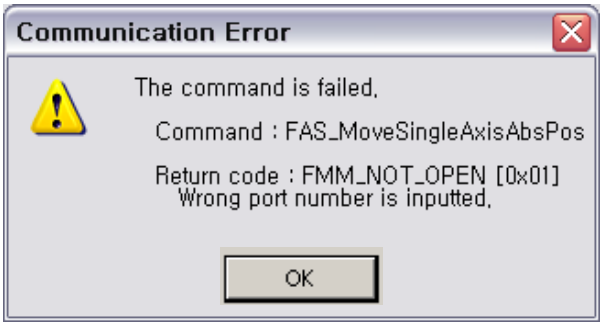
(2) The following table shows return values included commonly in all libraries and these functions offer to check the result (communication status, running status) judged by the drive. These functions are available for using library(DLL) and protocol.

Item	Description	Returned	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The drive cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.
Operation Error	FMP_ RUNFAIL	133	The motor is already running or not prepared for running.
	FMP_RESETFAIL	134	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	135	An alarm has occurred.
	FMP_SERVOONFAIL2	136	The motor is under Emergency Stop.
	FMP_SERVOONFAIL3	137	'ServoON' signal is already assigned to input pin.
Connection Error	FMP_PACKETERROR	130	Protocol level error occurs in packet that Drive's received.
	FMP_PACKETCRCERROR	170	CRC value is not correct in packet that Drive's received.

2-2. Communication Status Window

Above communication status is divided by 3 groups.

(1) Communication Error



FMM_NOT_OPEN,

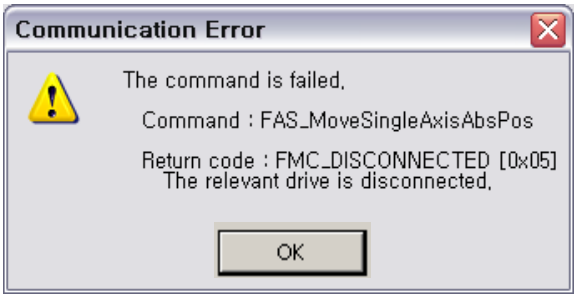
COM Port is not connected. (This error cannot be occurred in GUI.)

FMM_INVALID_PORT_NUM,

COM Port number is not exist. Checking the 'Device Manager' window in Window OS. (This error cannot be occurred in GUI.)

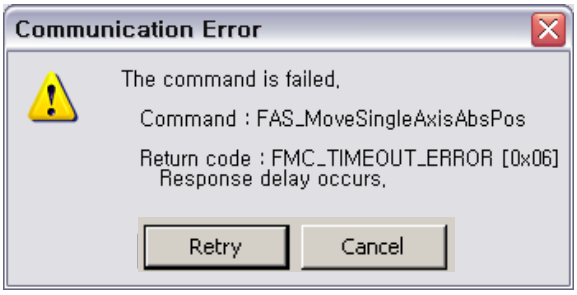
FMM_INVALID_SLAVE_NUM,

Slave number is not exist. Checking the ID value of the drive.
(This error cannot be occurred in GUI.)



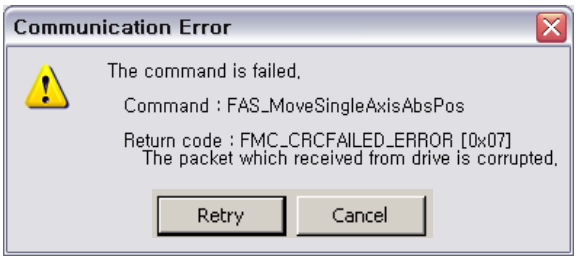
FMC_DISCONNECTED = 5,

COM Port is disconnected during communication. Checking the communication cable
Or Power of the drive.



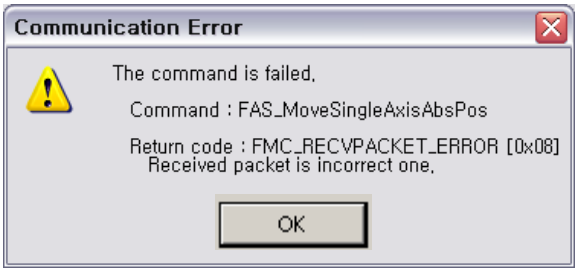
FMC_TIMEOUT_ERROR,

There is no response from the drive.



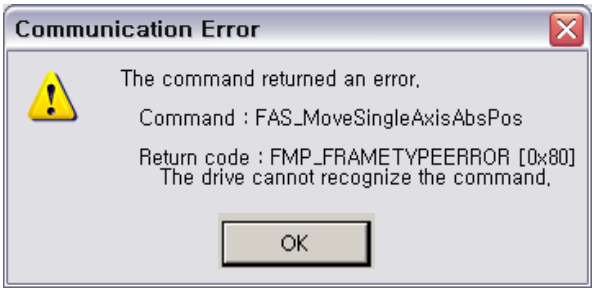
FMC_CRCFAILED_ERROR,

CRC value of communication packet from the drive is not correct. Checking the
Possibility of noise on communication cable.



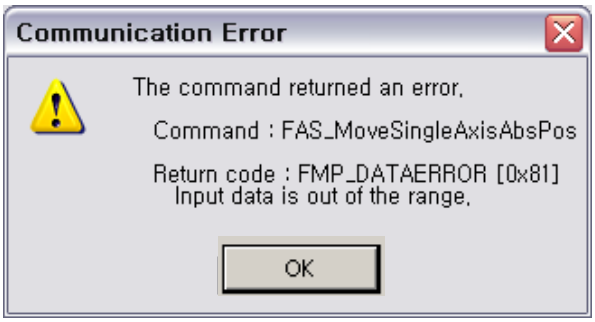
FMC_RECVPACKET_ERROR,

The length of received packet is not correct. Checking the possibility of noise on communication cable.



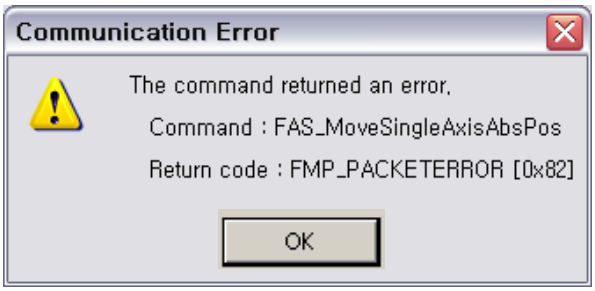
FMP_FRAMETYPEERROR = 0x80,

Drive do not recognize the command or wrong command is sent.
Checking the command value that you want to send to the drive.



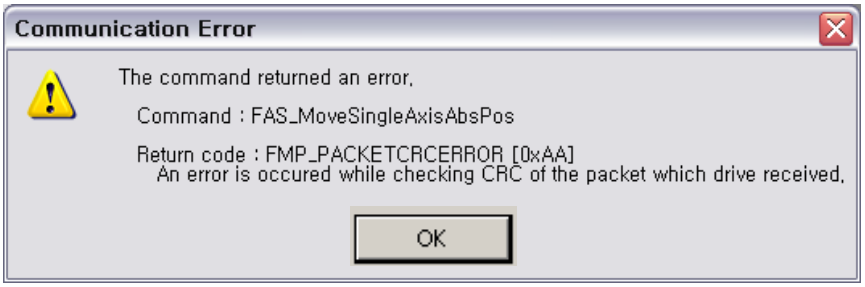
FMP_DATAERROR,

The value of the sent data is out of the proper range of the drive.
Checking the value that you want to send to the drive.



FMP_PACKETERROR,

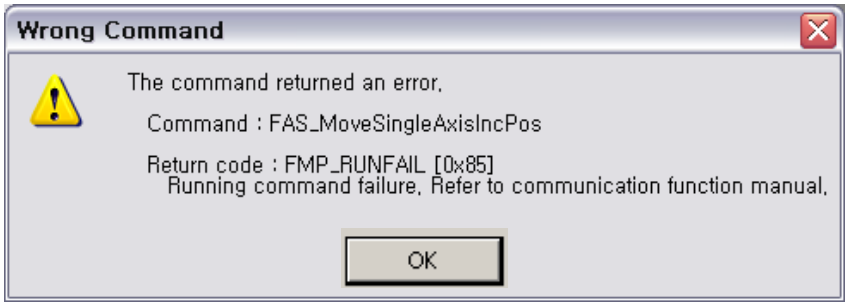
The length of received packet on drive is not correct. Checking the possibility of noise on communication cable.


Communication Error
The command returned an error.
Command : FAS_MoveSingleAxisAbsPos
Return code : FMP_PACKETCRCERROR [0xAA]
An error is occured while checking CRC of the packet which drive received.
OK

FMP_PACKETCRCERROR = 0xAA,

The incorrect CRC value of packet sent to the drive.

(2) Wrong Command


Wrong Command
The command returned an error.
Command : FAS_MoveSingleAxisIncPos
Return code : FMP_RUNFAIL [0x85]
Running command failure. Refer to communication function manual.
OK

FMP_RUNFAIL = 0x85,

Fail on motion command : Tried to new motion under following status.

-. The motor is already running

-. The motor is under stop command

-. Servo OFF status

-. Try to Z-pulse Origin without external encoder (for only Ezi-STEP)

Wrong Command
The command returned an error.
Command : FAS_ServoAlarmReset
Return code : FMP_RESETFAIL [0x86]
Running command failure. Refer to communication function manual.
OK

FMP_RESETFAIL,

Fail on reset command : The motor can not reset on next status.

-. Servo ON status

-. Already 'Reset' status by external input signal.

FMP_SERVOONFAIL1,

Wrong 'Servo ON' command during Alarm status.

FMP_SERVOONFAIL2,

Wrong 'Servo ON' command during E-Stop status.

Wrong Command
The command returned an error.
Command : FAS_ServoEnable
Return code : FMP_SERVOONFAIL3 [0x89]
The Servo-On signal is assigned at external input.
OK

FMP_SERVOONFAIL3,

'Servo ON' Signal is assigned to external input pin. In this case Servo ON command by DLL library does not work.

(3) Command Execution Error

Command Execution Failure
The command returned an error.
Command : FAS_PosTableWriteItem
Return code : FMM_POSTABLE_ERROR [0x09]
An error occurs while the motor accesses to the position table.
OK

FMM_POSTABLE_ERROR,

The execution of DLL library for 'Position Table' is failed.

2－3. Drive Link Function

Function Name	Description
FAS_Connect	The drive tries to connect communication with the drive module: When it is successfully connected, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_Close	The drive tries to disconnect communication with the drive module.
FAS_GetSlaveInfo	The drive reads drive type and program version: Drive type and version information will be returned.
FAS_IsSlaveExist	Check the existence of the relevant drive: When it exists, TRUE will be returned. Otherwise, FALSE will be returned.

FAS_Connect

FAS_Connect is the function of connecting Ezi-SERVO Plus-R.

Syntax

```
BOOL FAS_Connect(
    BYTE nPortNo,
    DWORD dwBaud
);
```

Parameters

nPortNo
Select a serial port number to be connected.

dwBaud
Input the Baudrate of the serial port.

Return Value

When it is successfully connected, TRUE will be returned. Otherwise, FALSE will be returned.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcInit()
{
        BYTE nPortNo = 1;  // COMM Port Number
        DWORD dwBaudrate = 115200;  // Baudrate. (Be variable by setting)
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        DRIVE_INFO infoDrv;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, dwBaudrate) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        if (FAS_IsSlaveExist(nPortNo, iSlaveNo) == FALSE)
        {
                // There is no relevant slave number.
                // Check the slave number of Ezi-STEP Plus-R.
                return;
        }

        nRtn = FAS_GetSlaveInfo(nPortNo, iSlaveNo, &infoDrv);
        if (nRtn != FMM_OK)
        {
                // Command has not been performed properly.
                // Refer to ReturnCodes_Define.h.
        }

        printf("Port : %d (Slave %d) Wn", nPortNo, iSlaveNo);
        printf("WtType : %d Wn", infoDrv.nDriveType);
        printf("Wt%s   Ver.%d.%d.%d.%d   Wn",   infoDrv.sVersion,   infoDrv.nVersionNo[0],
infoDrv.nVersionNo[1], infoDrv.nVersionNo[2], infoDrv.nVersionNo[3]);
```

```
        // Disconnect.
        FAS_Close(nPortNo);
    }
```

See Also

FAS_Close

FAS_Close

To disconnect the serial port being used

Syntax

```
void FAS_Close(
    BYTE nPortNo
);
```

Parameters

nPortNo

Port number to disconnected.

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

FAS_GetSlaveInfo

To get the information string of the relevant drive

Syntax

```
int FAS_GetSlaveInfo(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DRIVE_INFO* lpDriveInfo
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant drive
lpDriveInfo
Informations of relevant drive

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_IsSlaveExist

Check connection status of the drive.

Syntax

```
BOOL FAS_IsSlaveExist(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant drive

Return Value

TRUE : The drive is connected.
FALSE : The drive is disconnected.

Remarks

This function is provided from the library only and it is inapplicable to the protocol program mode.

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

2－4. Parameter Control Function

Function Name	Description
FAS_SaveAllParameters	Save current status of parameters to the ROM: Even after the drive is powered OFF, parameters related to operating speed, acceleration/deceleration time, and origin return need to be preserved.
FAS_SetParameter	Save designated parameter to the RAM: Specific parameter is saved.
FAS_GetParameter	Read designated parameter from the RAM: Specific parameter is read.
FAS_GetROMParameter	Read designated parameter from the ROM: Specific parameter is read from the ROM.

FAS_SaveAllParameters

Save all edited parameters up to now and assigned I/O signals to the ROM area.

Syntax

```
Int FAS_SaveAllParameters(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive

iSlaveNo
Slave number of relevant drive

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Parameter values set to 'FAS_SetIOAssignMap' library as well as current parameter values are saved in the ROM.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcModifyParameter()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        long lParamVal;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check Axis Start Speed Parameter.
        nRtn = FAS_GetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &lParamVal);
        if (nRtn != FMM_OK)
        {
                // Command has not been performed properly.
                // Refer to ReturnCodes_Define.h.
                _ASSERT(FALSE);
        }
        else
        {
                // Parameter value saved in Ezi-STEP Plus-R.
                printf("Parameter [before] : Start Speed = %d ₩n", lParamVal);
        }
```

```
        // Change Axis Start Speed parameter as 200 then read it again.
        nRtn = FAS_SetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, 200);
        _ASSERT(nRtn == FMM_OK);    // You have to check if the command didn't execute
correctly.

        nRtn = FAS_GetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &lParamVal);
        _ASSERT(nRtn == FMM_OK);
        printf("Parameter [after] : Start Speed = %d ₩n", lParamVal);


        // Check the value saved in the ROM.
        nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &lParamVal);
        _ASSERT(nRtn == FMM_OK);    // You have to check if the command didn't execute
correctly.
        printf("Parameter [ROM] : Start Speed = %d ₩n", lParamVal);


        // Edit the parameter value then save it in the ROM.
        nRtn = FAS_SetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, 100);
        _ASSERT(nRtn == FMM_OK);    // You have to check if the command didn't execute
correctly.

        nRtn = FAS_SaveAllParameters(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_GetROMParameter

FAS_SetParameter

Edit the relevant parameter value and then save it to the RAM.

Syntax

```
int FAS_SetParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long lParamValue
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant drive
iParamNo
Parameter number to be edited
lParamValue
Parameter value to be edited

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates for only one parameter designated.
Parameters in the drive are saved to 2 memory areas. That is, when power is off, the ROM saves parameters permanently. When power is on, parameters in the ROM are copied to the DSP RAM and used. When the user changes parameters, it changes not parameters in the ROM but parameter in the RAM. This function is to set the parameter number designated from the RAM to the relevant value.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_GetParameter

FAS_GetParamater

To call specific parameter value of the drive

Syntax

```
int FAS_GetParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long* lParamValue
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant drive
iParamNo
Parameter number to be brought
lParamValue
Parameter values

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates for only one parameter designated.
Parameters in the drive are saved to 2 memory areas. That is, when power is off, the ROM saves parameters permanently. When power is on, parameters in the ROM are copied to the DSP RAM and used. When the user changes parameters, it changes not parameters in the ROM but parameter in the RAM. This function reads the parameter number designated to the RAM.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SetParameter

FAS_GetROMParameter

To call parameters saved in the ROM

Syntax

```
int FAS_GetROMParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long* lROMParam
);
```

Parameters

nPortNo
Port number of relevant drive

iSlaveNo
Slave number of relevant drive

iParamNo
Parameter number to be brought

lROMParam
Parameter values saved in the ROM

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

To call parameter values saved in the ROM
Even though this function runs, the value in the RAM is not changed. For this, run FAS_SetParameter.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SaveAllParameters

2－5. Servo Control Function

Function Name	Description
FAS_ServoEnable	Servo ON/OFF of the designated drive
FAS_ServoAlarmReset	Release alarm of the drive generated alarm. Troubleshoot root cause of the alarm prior to use this function.

FAS_ServoEnable

Servo ON/OFF of the drive

Syntax

```
int FAS_ServoEnable(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BOOL bOnOff
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant
bOnOff
Enable or Disable.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

The given time is required until Servo ON flag in the axis status turns on after enable.

After 'ServoON' is executed, the 'Command Position' value will change as same as 'Actual Position' value to remove 'Position Error'.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void func10()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        DWORD dwInput, dwOutput;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check I/O input.
        nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
        _ASSERT(nRtn == FMM_OK);
        if (dwInput & STEP_IN_BITMASK_LIMITP)
        {
                // Limit + input is ON.
        }

        if (dwInput & STEP_IN_BITMASK_USERIN0)
        {
                // User Input 0 is ON.
        }
```

```
        // Turning ON 'Clear Position' and 'User Input 1' inputs and turning off 'Jog +' input.
        nRtn = FAS_SetIOInput(nPortNo, iSlaveNo, STEP_IN_BITMASK_CLEARPOSITION |
STEP_IN_BITMASK_USERIN1, STEP_IN_BITMASK_PJOG);
        _ASSERT(nRtn == FMM_OK);

        // Check I/O output.
        nRtn = FAS_GetIOOutput(nPortNo, iSlaveNo, &dwOutput);
        _ASSERT(nRtn == FMM_OK);
        if (dwOutput & STEP_OUT_BITMASK_USEROUT0)
        {
                // User Output 0 is ON.
        }

        // Turn off User Output 1 and 2 signals.
        nRtn = FAS_SetIOOutput(nPortNo, iSlaveNo, 0, STEP_OUT_BITMASK_USEROUT1 |
STEP_OUT_BITMASK_USEROUT2);
        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_ServoAlarmReset

FAS_ServoAlarmReset

To send AlarmReset command

Syntax

```
int FAS_ServoAlarmReset(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive
iSlaveNo
Slave number of relevant drive

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Before sending this command, troubleshoot root cause of the alarm.
For alarm cause, refer to 'User Manual_Text'.

Example

Refer to 'FAS_ServoEnable' library

See Also

FAS_ServoEnable

2－6. Control I/O Function

Function Name	Description
FAS_SetIOInput	To set the input signal level of the control input port : Set input signal [ON] or [OFF] status.
FAS_GetIOInput	To read the current input signal status of the control input port : The signal status returns by bit for each input signal.
FAS_SetIOOutput	To set the output signal level of the control input port : Set output signal [ON] or [OFF] status.
FAS_GetIOOutput	To read the current input signal status of the control output port : The signal status returns by bit for each output signal.
FAS_GetInputAssignMap	To read the Input pin of setting status of the CN1 port : The setting status for each 9 variable signals returns by bit to the Input port.
FAS_GetOutputAssignMap	To read the Output pin of setting status of the CN1 port : The setting status for each 9 variable signals returns by bit to the Output port.
FAS_SetInputAssignMap	To assign the control Input signal to CN1 port pin and also set the signal level : Setting for each 9 variable signals is assigned to the Input port.
FAS_SetOutputAssignMap	To assign the control Output signal to CN1 port pin and also set the signal level : Setting for each 9 variable signals is assigned to the Output port.
FAS_IOAssignMapReadROM	To load the pin of setting status of CN1 port from ROM area to RAM area.

FAS_SetIOInput

To set I/O input. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_SetIOInput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
dwIOSetMask
Input bitmask value to be set
dwIOCLRMask
Input bitmask value to be cleared

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Be careful that dwIOSetMask bit and dwIOCLRMask bit are not duplicated.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcIO()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        DWORD dwInput, dwOutput;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check I/O input.
        nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
        _ASSERT(nRtn == FMM_OK);
        if (dwInput & STEP_IN_BITMASK_LIMITP)
        {
                // Limit + input is ON.
        }
```

```
		if (dwInput & STEP_IN_BITMASK_USERIN0)
		{
			// User Input 0 is ON.
		}

		// Turning ON 'Clear Position' and 'User Input 1' inputs and turning off 'Jog +' input.
		nRtn = FAS_SetIOInput(nPortNo, iSlaveNo, STEP_IN_BITMASK_CLEARPOSITION |
	STEP_IN_BITMASK_USERIN1, STEP_IN_BITMASK_PJOG);
		_ASSERT(nRtn == FMM_OK);

		// Check I/O output.
		nRtn = FAS_GetIOOutput(nPortNo, iSlaveNo, &dwOutput);
		_ASSERT(nRtn == FMM_OK);
		if (dwOutput & STEP_OUT_BITMASK_USEROUT0)
		{
			// User Output 0 is ON.
		}

		// Turn off User Output 1 and 2 signals.
		nRtn = FAS_SetIOOutput(nPortNo, iSlaveNo, 0, STEP_OUT_BITMASK_USEROUT1 |
	STEP_OUT_BITMASK_USEROUT2);
		_ASSERT(nRtn == FMM_OK);

		// Disconnect.
		FAS_Close(nPortNo);
	}
```

See Also

FAS_GetIOInput

FAS_GetIOInput

To read I/O input values. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_GetIOInput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwIOInput
);
```

Parameters

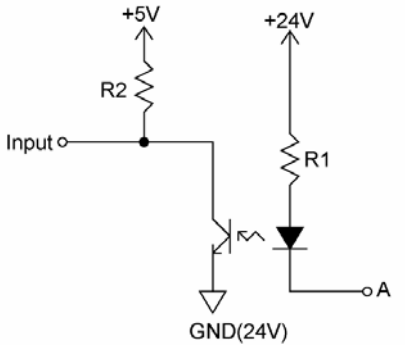
nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
dwIOInput
Parameter pointer where input values will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

There are 12 input pins in EziSERVO PlusR. The user can select and use 9 input pins of them. This function can read the input port status as 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



If voltage from an external input, is 24V at Port A, the input is recognized to 5V(High).

Example

Refer to 'FAS_SetIOInput' library.

See Also

FAS_SetIOInput

FAS_SetIOOutput

To set I/O output values. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_SetIOOutput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

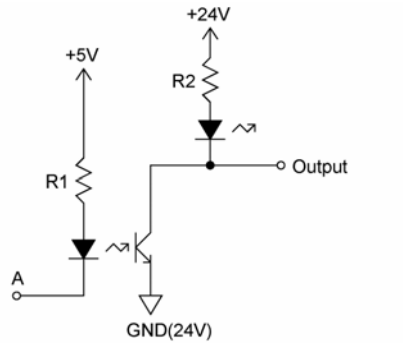
nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
dwIOSetMask
Output bitmask value to be set (ON status)
dwIOCLRMask
Output bitmask value be cleared (OFF status)

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

There are 10 output pins in EziSERVO PlusR. The user can select and use 9 output pins of them.



When output data is '1', Port A becomes 0V. When it is '0', Port A becomes +5V.

Be careful that dwIOSetMask bit and dwIOCLRMask bit are not duplicated.

Example

Refer to FAS_SetIOInput.

See Also

FAS_GetIOOutput

FAS_GetIOOutput

To read I/O output values. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_GetIOOutput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwIOOutput
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

dwIOInput
Parameter pointer where the output value will be saved.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_SetIOInput' library

See Also

FAS_SetIOOutput

FAS_GetInputAssignMap

To read Input Assign Map. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_GetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iInPinNo,
    BYTE* nInLogicNo,
    BYTE* bLevel
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

iInPinNo
Input pin number to be read

nInLogic
Parameter pointer where the logic value assigned to a relevant pin will be saved

bLevel
Parameter pointer where the active level of relevant logic will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

For nInLogicNo, refer to 'MOTION_xxxxx_DEFINE.h' according to drive type.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcIOAssign()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        BYTE iPinNo;
        BYTE nLogicNo;
        BYTE bLevel;
        BYTE i;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check assigned information of input pin.
        for (i=0; i</*Input Pin Count*/12; i++)
        {
```

```
                    nRtn = FAS_GetInputAssignMap(nPortNo, iSlaveNo, i, &nLogicNo, &bLevel);
                    _ASSERT(nRtn == FMM_OK);

                    if (dwLogicMask != IN_LOGIC_NONE)
                            printf("Input Pin %d : Logic No. %d (%s)₩n", i, nLogicNo, ((bLevel
== LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
                    else
                            printf("Input Pin %d : Not assigned₩n", i);
            }

            // Assign E-Stop Logic (Low Active) to input pin 3.
            iPinNo = 3;         // 0 ~ 11 value is available (Caution : 0 ~ 2 is fixed.)
            nRtn  =  FAS_SetInputAssignMap(nPortNo,  iSlaveNo,  iPinNo,  STEP_IN_LOGIC_ESTOP,
LEVEL_LOW_ACTIVE);
            _ASSERT(nRtn == FMM_OK);

            // Check assign information of output pin.
            for (i=0; i<10/*Output Pin Count*/; i++)
            {
                    nRtn = FAS_GetOutputAssignMap(nPortNo, iSlaveNo, i, &nLogicNo, &bLevel);
                    _ASSERT(nRtn == FMM_OK);

                    if (dwLogicMask != OUT_LOGIC_NONE)
                            printf("Output Pin %d : Logic No. %d (%s)₩n", i, nLogicNo, ((bLevel
== LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
                    else
                            printf("Output Pin %d : Not assigned₩n", i);
            }

            // Assign ALARM Logic (High Active) to output pin 9.
            iPinNo = 9;         // 0 ~ 9 value is available (Caution : 0 is fixed to COMPOUT.)
            nRtn  =  FAS_SetOutputAssignMap(nPortNo,  iSlaveNo,  iPinNo,  SERVO_OUT_LOGIC_ALARM,
LEVEL_HIGH_ACTIVE);
            _ASSERT(nRtn == FMM_OK);

            // Disconnect.
            FAS_Close(nPortNo);
    }
```

See Also

FAS_GetOutputAssignMap, FAS_SetInputAssignMap, FAS_SetOutputAssignMap

FAS_GetOutputAssignMap

To read Output Assign Map. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_GetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iOutPinNo,
    BYTE* nOutLogicNo,
    BYTE* bLevel
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
iOutPinNo
Output pin number to be read
nOutLogicNo
Parameter pointer where the logic value assigned to a relevant pin will be saved
bLevel
Parameter pointer where the active level of relevant logic will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

For nInLogicNo, refer to 'MOTION_xxxxx_DEFINE.h' according to drive type.

Example

Refer to 'FAS_GSetIOAssignMap' library

See Also

FAS_GetInputAssignMap, FAS_SetInputAssignMap, FAS_SetOutputAssignMap

FAS_ SetInputAssignMap

To set Input Assign Map. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_SetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iInPinNo,
    BYTE nInLogicNo,
    BYTE bLevel
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
iInPinNo
Input Pin number to be read
nInLogicNo
Logic value to be assigned to the relevant pin
bLevel
Active Level value of the relevant logic

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : Designated iIOPinNo or nIOLogic value is out of range.

Remarks

To save current setting values to the ROM memory, 'FAS_SaveAllParameters' library should be run.

Example

Refer to 'FAS_GetInputAssignMap' library

See Also

FAS_GetInputAssignMap, FAS_GetOutputAssignMap, FAS_SetOutputAssignMap

FAS_ SetOutputAssignMap

To set Output Assign Map. For more information, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_SetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iOutPinNo,
    BYTE nOutLogicNo,
    BYTE bLevel
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
iOutPinNo
Input Pin number to be read
nOutLogicNo
Logic value to be assigned to the relevant pin
bLevel
Active Level value of the relevant logic

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : Designated iIOPinNo or nIOLogic value is out of range.

Remarks

To save current setting values to the ROM memory, 'FAS_SaveAllParameters' library should be run.

Example

Refer to 'FAS_GetInputAssignMap' library

See Also

FAS_GetInputAssignMap, FAS_GetOutputAssignMap, FAS_SetOutputAssignMap

FAS_IOAssignMapReadROM

To load the status of CN1 assignment I/O setting status and signal level in ROM area

Syntax

```
int FAS_PosTableReadROM(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

See Also

FAS_ GetIOAssignMap

2－7. Position Control Function

Function Name	Description
FAS_SetCommandPos	To set the command position value
FAS_SetActualPos	To set the current position to the actual position value
FAS_GetCommandPos	To read the current command position value
FAS_GetActualPos	To read the current actual position value
FAS_GetPosError	To read the difference between the actual position value and the command position value
FAS_GetActualVel	To read the actual running speed value while the motor is moving
FAS_ClearPosition	To set the command position and actual position value to ‘0’

FAS_SetCommandPos

To set the command position value of the motor

Syntax

```
int FAS_SetCommandPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lCmdPos
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lCmdPos
Command position value to be set.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

The user sets the position command (pulse output counter) value.
This function is generally used when the user sets the current position to coordinates that customer wants.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcClearPosition()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Initialize Command Position and Actual Position values to 0.
        nRtn = FAS_SetCommandPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);
        nRtn = FAS_SetActualPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_SetActualPos

FAS_SetActualPos

To set the actual position value to the motor

Syntax

```
int FAS_SetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lActPos
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lActPos
Actual position value to be set.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

The user sets the encoder feedback counter value to the value what customer wants.

Example

Refer to 'FAS_GetActualPos' library.

See Also

FAS_SetCommandPos

FAS_GetCommandPos

To read the command position of the current motor

Syntax

```
int FAS_GetCommandPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lCmdPos
);
```

Parameters

nPortNo
Port number of relevant drive

iSlaveNo
Slave number of relevant drive

lCmdPos
Parameter pointer that command position value will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

To read the position command (pulse output counter) value.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcDisplayStatus()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        long lValue;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check position information of Ezi-STEP Plus-R.
        nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("CMDPOS : %d Wn", lValue);
        nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("ACTVEL : %d Wn", lValue);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_GetActualPos

FAS_GetActualPos

To read the actual position value of the motor

Syntax

```
int FAS_GetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActPos
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lActPos
Parameter pointer where the actual position value will be saved.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

When the user decides the motor position and checks its actual position, this function is generally used.

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos

FAS_GetPosError

To read the position error of the motor

Syntax

```
int FAS_GetPosError(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lPosErr
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lPosErr
Parameter pointer where the position error value will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos,
FAS_GetActualPos

FAS_GetActualVel

To read the actual velocity of the motor

Syntax

```
int FAS_GetActualVel(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActVel
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lActVel
Parameter pointer where the actual velocity value will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_ClearPosition

To set the command position value and actual position value of the motor to '0'

Syntax

```
int FAS_ClearPosition(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

The user sets the position command (pulse output counter) value.
This function is generally used when the user sets the current position to initial values.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcClearPosition()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Initialize Command Position and Actual Position values to 0.
        nRtn = FAS_ClearPosition(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_SetActualPos

2－8. Drive Status Control Function

Function Name	Description
FAS_GetIOStatus	To read control I/O status, running status Flag value : The current input status value, the output setting status value, and the running status Flag value will be returned.
FAS_GetMotionStatus	To read the current running progress status and its PT number : The command position value, the actual position value, the speed value will be returned.
FAS_GetAllStatus	To read all status includes the current I/O status at one time : This function is to combine 'FAS_GetIOAxisStatus' function and 'FAS_GetMotionStatus' function.
FAS_GetAxisStatus	To read the running status Flag value of the relevant drive
FAS_GetRunPTStatus	To read the PositionTable number that is running.

FAS_GetIOStatus

To read I/O Input and Output values of the relevant drive, and the motor Axis Status value

Syntax

```
int FAS_GetIOAxisStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    ULONGLONG* ulnStatus,
    DWORD* dwOutStatus
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
ulnStatus
Parameter pointer where the I/O input value will be saved.
dwOutStatus
Parameter pointer where the I/O output value will be saved.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetMotionStatus

To read the motion status of current motor at one time

Syntax

```
int FAS_GetMotionStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lCmdPos,
    long* lActPos,
    long* lPosErr,
    long* lActVel,
    WORD* wPosItemNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lCmdPos
Parameter pointer where the command position value will be saved
lActPos
Parameter pointer where the actual position value will be saved.
lPosErr
Parameter pointer where the position error value will be saved
lActVel
Parameter pointer where the actual velocity value will be saved
wPosItemNo
Parameter pointer where current running item number in the Position Table will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAllStatus

To read I/O Input and Output values of the relevant drive, the motor Axis Status, the motor motion status value at one time

Syntax

```
int FAS_GetAllStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwInStatus,
    DWORD* dwOutStatus,
    DWORD* dwAxisStatus,
    long* lCmdPos,
    long* lActPos,
    long* lPosErr,
    long* lActVel,
    WORD* wPosItemNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
dwInStatus
Parameter pointer where the I/O input value will be saved.
dwOutStatus
Parameter pointer where the I/O output value will be saved.
dwAxisStatus
Parameter pointer where the axis status value of the relevant motor will be saved
lCmdPos
Parameter pointer where the command position value will be saved
lActPos
Parameter pointer where the actual position value will be saved
lPosErr
Parameter pointer where the position error value will be saved
lActVel
Parameter pointer where the actual velocity value will be saved
wPosItemNo
Parameter pointer where current running item number in the Position Table will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAxisStatus
FAS_GetMotionStatus

FAS_GetAxisStatus

To read the motor Axis Status value. For status Flag, refer to '1-2. Structure of Frame Type'.

Syntax

```
int FAS_GetAxisStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwAxisStatus
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

dwAxisStatus
Parameter pointer where the axis status value of the relevant motor

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetRunPTStatus

To read the Position Table number that running currently.

Syntax

```
int FAS_GetRunPTStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD* wPTStatus
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
wPTStatus
Variable pointer that can be save the number of PT running.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : 아직 Board를 연결하기 전 입니다.
FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort는 존재하지 않습니다.
FMM_INVALID_SLAVE_NUM : 해당 Port에 iSlaveNo의 Slave는 존재하지 않습니다.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library

See Also

2－9. Running Control Function

Function Name	Description
FAS_MoveStop	Stop the motor in running with deceleration.
FAS_EmergencyStop	Stop the motor in running immediately without deceleration.
FAS_MoveOriginSingleAxis	Start operation to return origin.
FAS_MoveSingleAxisAbsPos	The motor moves as much as the given absolute position value.
FAS_MoveSingleAxisIncPos	The motor moves as much as the given incremental position value.
FAS_MoveToLimit	The motor moves up to the position where the limit sensor is detected.
FAS_MoveVelocity	The motor moves to the given velocity and direction: This function is available to Jog motion.
FAS_PositionAbsOverride	Change the target absolute position value [pulse] of the motor in running.
FAS_PositionIncOverride	Change the target incremental position value [pulse] of the motor in running.
FAS_VelocityOverride	Change the running velocity value [pps] of the motor in running.
FAS_AllMoveStop	Stop all motors connected in same port with deceleration.
FAS_AllEmergencyStop	Stop all motors connected in same port immediately without deceleration.
FAS_AllMoveOriginSingleAxis	Start operation to return all motors in same port to origin position.
FAS_AllMoveSingleAxisAbsPos	All motors that connected in same port moves as much as the given absolute position value.
FAS_AllMoveSingleAxisIncPos	All motors that connected in same port moves as much as the given incremental position value.
FAS_MoveLinearAbsPos	More than 2 motors that connected in same port Linear Interpolation moves as much as the given absolute position value.
FAS_MoveLinearIncPos	More than 2 motors that connected in same port Linear Interpolation moves as much as the given incremental position value.
FAS_MoveSingleAxisAbsPosEx	The motor moves as much as the given absolute position value with custom accel/decel time value.
FAS_MoveSingleAxisIncPosEx	The motor moves as much as the given incremental position value with custom accel/decel time value.
FAS_MoveVelocityEx	The motor moves to the given velocity and direction: This function is available to Jog motion with custom accel/decel time value.

FAS_MoveStop

To stop the motor

Syntax

```
int FAS_MoveStop(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_EmergencyStop

To stop the motor without deceleration

Syntax

```
int FAS_EmergencyStop(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

This function does not include deceleration phase. So, the user must be careful so that the machine cannot be impacted.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveOriginSingleAxis

To search the origin of system. For more information, refer to 'User Manual_Text 9.3 Origin Return'.

Syntax

```
int FAS_MoveOriginSingleAxis(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveSingleAxisAbsPos

To move the motor to the absolute coordinate value

Syntax

```
int FAS_MoveSingleAxisAbsPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lAbsPos,
    DWORD lVelocity
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lAbsPos
Absolute coordinate where position to move
lVelocity
Velocity when the motor moves

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcMove()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        DWORD dwAxisStatus, dwInput;
        EZISTEP_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Check error status.
        nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
        _ASSERT(nRtn == FMM_OK);
        stAxisStatus.dwValue = dwAxisStatus;

        //if (dwAxisStatus & 0x00000001)
        if (stAxisStatus.FFLAG_ERRORALL)
```

```
                FAS_StepAlarmReset(nPortNo, iSlaveNo);

        // Check input status.
        nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
        _ASSERT(nRtn == FMM_OK);

        if (dwInput & (STEP_IN_LOGIC_STOP | STEP_IN_LOGIC_PAUSE | STEP_IN_LOGIC_ESTOP))
                FAS_SetIOInput(nPortNo,     iSlaveNo,     0,     STEP_IN_LOGIC_STOP     |
STEP_IN_LOGIC_PAUSE | STEP_IN_LOGIC_ESTOP);

        // Increase the motor to 15000 pulse.
        lIncPos = 15000;
        lVelocity = 30000;
        nRtn = FAS_MoveSingleAxisIncPos(nPortNo, iSlaveNo, lIncPos, lVelocity);
        _ASSERT(nRtn == FMM_OK);

        // Stand by until motion command is completely finished.
        do
        {
                Sleep(1);

                nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // Move the motor to '0'.
        lAbsPos = 0;
        lVelocity = 20000;
        nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
        _ASSERT(nRtn == FMM_OK);

        // Stand by until motion command is completely finished
        do
        {
                Sleep(1);

                nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_MoveSingleAxisIncPos

To move the motor to the incremental coordinate value

Syntax

```
int FAS_MoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lIncPos,
    DWORD lVelocity
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lIncPos
Incremental coordinate where position to move
lVelocity
Velocity when the motor moves

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveToLimit

To give the motor a command to search the limit sensor

Syntax

```
int FAS_MoveToLimit(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iLimitDir
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lVelocity
Velocity when the motor moves
iLimitDir
Limit direction of the motor moves (0: -Limit, 1: +Limit)

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveVelocity

To move the motor to the relevant direction and velocity. This function is available for Jog motion.

Syntax

```
int FAS_MoveVelocity(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iVelDir
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lVelocity
Velocity when the motor moves
iVelDir
Direction when the motor moves (0: -Jog, 1: +Jog)

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionAbsOverride

To change the absolute position value set while the motor moves to the absolute position

Syntax

```
int FAS_PositionAbsOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lOverridePos

Absolute coordinate position value to be changed

Return Value

FMM_OK : Command has been successfully performed.

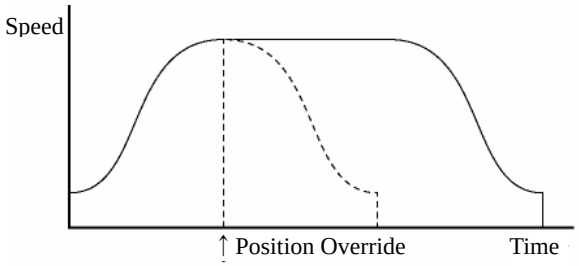
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

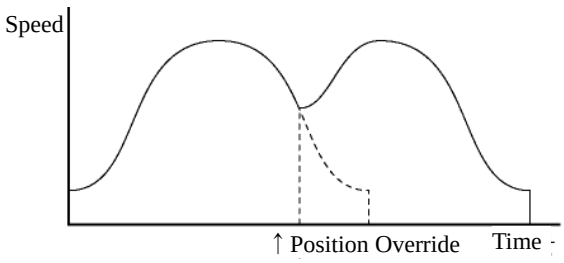
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

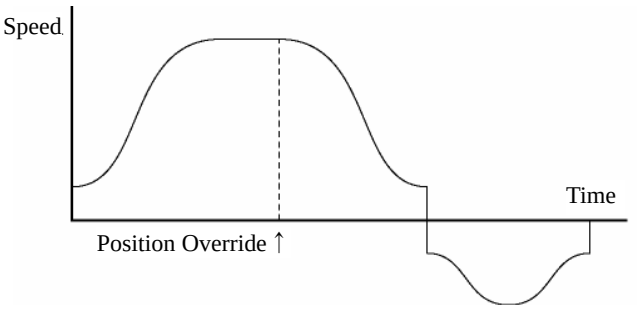
1) If the target position is set to the farther coordinate than the original target position while the motor moves under acceleration or constant velocity, the motor moves to the velocity pattern until then and stops the target position.



2) If the target position is changed while the motor is decelerated, it is again accelerated up to the constant velocity and then stops at the target position.



3) If the changed target position is set to the closer coordinate than the original target position, the motor once stops at the position before changed and then performs acceleration and deceleration to stop at the changed target position.



Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionIncOverride

FAS_PositionIncOverride

To change the incremental position value set while the motor moves to the incremental position

Syntax

```
int FAS_PositionIncOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lOverridePos
Incremental coordinate position value to be changed

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Refer to 'FAS_PositionAbsOverride' library.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionAbsOverride

FAS_VelocityOverride

To change the velocity set while the motor moves

Syntax

```
int FAS_VelocityOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity
);
```

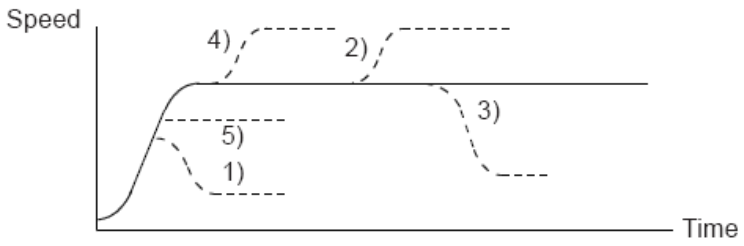
Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lVelocity
Velocity to be changed in [pps]

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks



1) In case of ((change speed) < (speed before change)), the motor reaches to the changed speed through acceleration/deceleration using a new velocity pattern.
5) In case of ((change speed) ≥ (speed before change)), the motor reaches to the changed speed through acceleration/deceleration without any new velocity pattern.
4) The motor reaches to the 'speed before change' without change of the velocity pattern and then it reaches to the 'change speed' by a new velocity pattern.
2),3) After acceleration/deceleration is finished, the motor reaches to the changed speed corresponding to the velocity pattern of the 'change speed'.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveStop

To stop all motors that connected in same port.

Syntax

```
int FAS_AllMoveStop(
    BYTE nPortNo
);
```

Parameters

nPortNo
Port number of relevant drive.

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllEmergencyStop

To stop all motors that connected in same port without deceleration

Syntax

```
int FAS_AllEmergencyStop(
    BYTE nPortNo
);
```

Parameters

nPortNo
Port number of relevant drive.

Return Value

No response

Remarks

This function does not include deceleration phase. So, the user must be careful so that the machine cannot be impacted.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveOriginSingleAxis

To search the origin of system for all motor that are connected in same port. For more information, refer to 'User Manual_Text 9.3 Origin Return'.

Syntax

```
int FAS_AllMoveOriginSingleAxis(
    BYTE nPortNo
);
```

Parameters

nPortNo
Port number of relevant drive.

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveSingleAxisAbsPos

To move all motors that connected in same port to the absolute coordinate value

Syntax

```
int FAS_AllMoveSingleAxisAbsPos(
    BYTE nPortNo,
    long lAbsPos,
    DWORD lVelocity
);
```

Parameters

nPortNo
Port number of relevant drive.

lAbsPos
Absolute coordinate of position to move

lVelocity
Velocity when the motor moves

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveSingleAxisIncPos

To move all motors that connected in same port to the incremental coordinate value

Syntax

```
int FAS_AllMoveSingleAxisIncPos(
    BYTE nPortNo,
    long lIncPos,
    DWORD lVelocity
);
```

Parameters

nPortNo
Port number of relevant drive.

lIncPos
Incremental coordinate of position to move

lVelocity
Velocity when the motor moves

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveLinearAbsPos / FAS_MoveLinearIncPos

To move(Linear Interpolation) more than 2 motors that connected in same port to the absolute /incremental coordinate value.

Syntax

```
int FAS_MoveLinearAbsPos(
    BYTE nPortNo,
    BYTE nNoOfSlaves,
    BYTE *iSlaveNo,
    long *lAbsPos,
    DWORD lFeedrate,
    WORD wAccelTime
);
```

Parameters

nPortNo
Port number of relevant drive.

nNoOfSlaves
Slave numbers for Linear motioning.

iSlaveNo
Array of Slave numbers.

lAbsPos
Array of position value for each slave. (in case of incremental : Distance)

lFeedrate
Speed value for motioning.

wAccelTime
Acceleration & deceleration time value.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

When this function is used, the 'Start speed' value must be designated to '1'.

FAS_MoveSingleAxisAbsPosEx

To move the motor to the absolute coordinate

Syntax

```
int FAS_MoveSingleAxisAbsPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lAbsPos,
    DWORD lVelocity,
    MOTION_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

lAbsPos
Absolute coordinate of position to move

lVelocity
Velocity when the motor moves

lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Refer to MOTION_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMoveEx()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        DWORD dwAxisStatus, dwInput;
        EZISERVO_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        MOTION_OPTION_EX opt = {0};
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port number may be wrong, or incorrect Baudrate.
                return;
        }

        // Moving motor with different acc/dec time
        lIncPos = 15000;
        lVelocity = 30000;
```

```
        opt.flagOption.BIT_USE_CUSTOMACCEL = 1;
        opt.flagOption.BIT_USE_CUSTOMDECEL = 1;

        opt.wCustomAccelTime = 50;
        opt.wCustomDecelTime = 200;

        nRtn = FAS_MoveSingleAxisIncPosEx(nPortNo, iSlaveNo, lIncPos, lVelocity, &opt);
        _ASSERT(nRtn == FMM_OK);

        // Waiting until motioning is done.
        do
        {
                Sleep(1);

                nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // Moving motor to position 0.
        lAbsPos = 0;
        lVelocity = 20000;
        nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
        _ASSERT(nRtn == FMM_OK);

        // Waiting until motioning is done.
        do
        {
                Sleep(1);

                nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_MoveSingleAxisIncPosEx

To move the motor to the Incremental coordinate

Syntax

```
int FAS_MoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lIncPos,
    DWORD lVelocity,
    MOTION_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lIncPos
Incremental coordinate of position to move
lVelocity
Velocity when the motor moves
lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

See Also

FAS_MoveVelocityEx

To move the motor to the relevant direction and velocity. This function is also available for Jog motion.

Syntax

```
int FAS_MoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iVelDir,
    VELOCITY_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
lVelocity
Velocity when the motor moves
iVelDir
Direction which the motor moves (0: -Jog, 1: +Jog)
lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Refer to VELOCITY_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMoveVelocityEx()
{
        BYTE nPortNo = 1;    // COMM Port Number
        BYTE iSlaveNo = 0;   // Slave No (0 ~ 15)
        long lVelocity;
        VELOCITY_OPTION_EX opt = {0};
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port number may be wrong, or incorrect Baudrate.
                return;
        }

        // Moving motor with different acc/dec time : FAS_MoveSingleAxisIncPosEx
        lVelocity = 30000;

        opt.flagOption.BIT_USE_CUSTOMACCDEC = 1;
        opt.wCustomAccDecTime = 300;
```

```
        nRtn = FAS_MoveVelocityEx(nPortNo, iSlaveNo, lVelocity, DIR_INC, &opt);
        _ASSERT(nRtn == FMM_OK);

        Sleep(5000);
        FAS_MoveStop(nPortNo, iSlaveNo);
    }
```

See Also

2－10. Position Table Control Function

Function Name	Description
FAS_PosTableReadItem	To read items of RAM area in the specific position table
FAS_PosTableWriteItem	To save specific all items of position table to RAM area
FAS_PosTableWriteROM	To save all of position table values to ROM area : Total 256 PT values are saved.
FAS_PosTableReadROM	To read position table values in ROM area : Total 256 PT values are read.
FAS_PosTableRunItem	The motor starts to run from the designated position table in sequence.
FAS_PosTableSingleRunItem	To perform command of a specific item in the position table
FAS_PosTableReadOneItem	To read items of RAM area in the specific one item of position table
FAS_PosTableWriteOneItem	To save specific item of specific position table to RAM area

FAS_PosTableReadItem

To read a specific item in the position table

Syntax

```
int FAS_PosTableReadItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
wItemNo
Item number to be read
lpItem
Item structure pointer which item value is saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcPosTable()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
        WORD wItemNo;
        ITEM_NODE nodeItem;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        // Read No.20 Position table value and edit the position value.
        wItemNo = 20;
        nRtn = FAS_PosTableReadItem(nPortNo, iSlaveNo, wItemNo, &nodeItem);
        _ASSERT(nRtn == FMM_OK);

        nodeItem.lPosition = 260000;          // Change the position value to 260000.
        nodeItem.wBranch = 23;                // Set next command to 23.
```

```
        nodeItem.wContinuous = 1;            // Next command should be connected without
deceleration.

        nRtn = FAS_PosTableWriteItem(nPortNo, iSlaveNo, wItemNo, &nodeItem);
        _ASSERT(nRtn == FMM_OK);

        // Call the value in the ROM regardless of edited position table data.
        nRtn = FAS_PosTableReadROM(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

        // Save edited position table data in the ROM.
        nRtn = FAS_PosTableWriteROM(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(nPortNo);
}
```

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

To edit specific items in the position table

Syntax

```
int FAS_PoslableWriteItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
wItemNo
Item number to be edited
lpItem
Item structure pointer to be edited

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMC_POSTABLE_ERROR : An error occurs while position table is being written.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Position Table data is saved to RAM / ROM area. This function activates to save data to RAM area. When power is off, data is deleted.

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteROM

To save all current position table items to ROM area

Syntax

```
int FAS_PosTableWriteROM(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMC_POSTABLE_ERROR : An error occurs while position table is being saved.

Remarks

Position table data is saved to RAM / ROM area. This function activates to save data to ROM area. Even though power is off, data is preserved.

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

To read position table items being saved in ROM area

Syntax

```
int FAS_PosTableReadROM(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMC_POSTABLE_ERROR : An error occurs while position table is being read.

Remarks

Example

See Also

FAS_PosTableWriteROM

FAS_PosTableRunItem

To perform command from a specific item in the position table

Syntax

```
int FAS_PosTableRunItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
wItemNo
Item number to start motion

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_GetAllStatus
FAS_MoveStop
FAS_EmergencyStop

FAS_PosTableSingleRunItem

To perform command of a specific item in the position table

Syntax

```
int FAS_PosTableSingleRunItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BOOL bNextMove,
    WORD wItemNo
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
bNextMove
Flag whether runs the next step or runs the step of wItemNo.
wItemNo
Item number to execute motion

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

If bNextMove value is 0, the item of 'wItemNo' will be executed. If bNextMove value is not 0, the 'JPTableNo' item of the item executed last time will be executed - whatever 'wItemNo' is. The 'JPTableNo' of the item executed last time is empty, or FAS_PosTableSingleRunItem hasn't executed , this function execute nothing when 'bNextMove' is not 0.

Example

See Also

FAS_PosTableRunItem
FAS_GetAllStatus
FAS_MoveStop
FAS_EmergencyStop

FAS_PosTableReadOneItem

To read specific item in the specific position table

Syntax

```
int FAS_PosTableReadOneItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    WORD wOffset,
    long* lPosItemVal
);
```

Parameters

nPortNo
Port number of relevant drive.
iSlaveNo
Slave number of relevant drive.
wItemNo
Item number to be read
wOffset
offset value which will be read from PT items. (Refer to '1-2-6. Position Table Item')
lPosItemVal
Parameter pointer where PT item data value will be saved

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableWriteOneItem

FAS_PosTableWriteOneItem

To edit specific item in the specific position table

Syntax

```
int FAS_PoslableWriteOneItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    WORD wOffset,
    long lPosItemVal
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

wItemNo
Item number to be edited

wOffset
offset value which will be save from PT items. (Refer to '1-2-6. Position Table Item')

lPosItemVal
PT item data value to be set

Return Value

FMM_OK : Command has been successfully performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMC_POSTABLE_ERROR : An error occurs while position table is being written.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

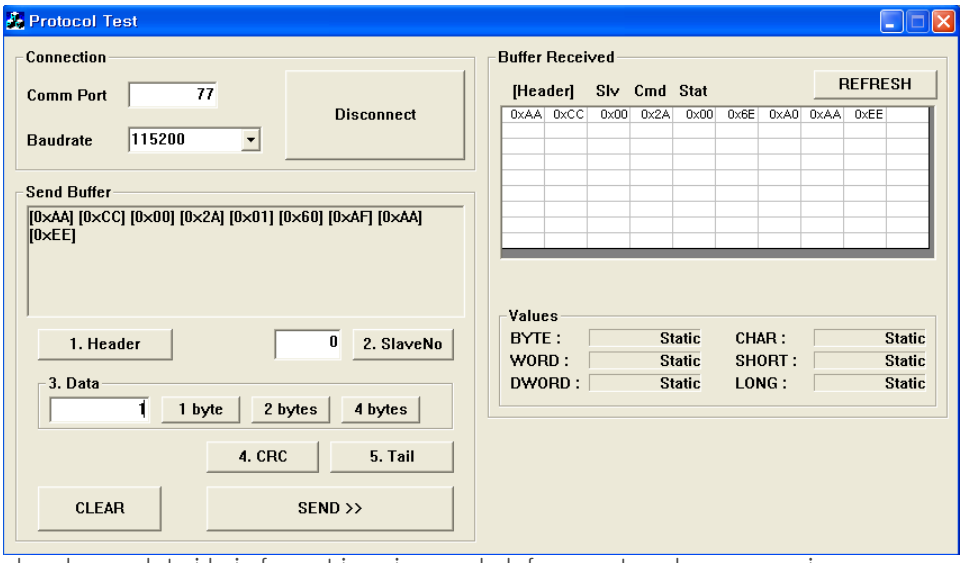
FAS_PosTableReadOneItem

3. Protocol test Program

Next window activates when you click

icon in User Program(GUI) installed folder.
Next test procedure will help you to understand the protocol programming.

(1)Servo ON/OFF command purpose of command



The header and tail information is needed for protocol programming.
Additionally Frame Data (Slave ID, Frame type, Data and CRC) is also needed in each one of protocol with header and tail.

1)Insert 'Comm Port' number and 'Baudrate', and click 'Connect' button.

2)Header : Click 'Header' and you can see '[0xAA][0xCC]' on 'Send Buffer' window.

3)Slave ID : Insert your connected slave number(above example is '0') and click 'SlaveNo'.

4) Sync No. : Insert some random value ans click '1byte'.
When you repeat the same command(Frame type) change this value.

5) Frame type : Select 'Frame type'.
You can find next table information in '1-2-1. Frame Type and Data Configuration' on UserManual(EziSERVO PlusR)_Communication Function about Servo ON/OFF command.

Frame type	DLL Library name	Data
42 (0x2A)	FAS_ServoEnable	Setting the Servo ON/OFF status. Sending : 1 byte 1 byte 0:OFF, 1:ON

Insert '42' in ① area and click '1 byte' because the size of Frame Type is 1 byte.

5)Data : To make Servo ON status, the data is '1' so Insert '1' in ① area and click '1 byte'.

6)CRC : Click 'CRC' and the automatically calculated result value(2 bytes) is displayed on 'Send Buffer' window.

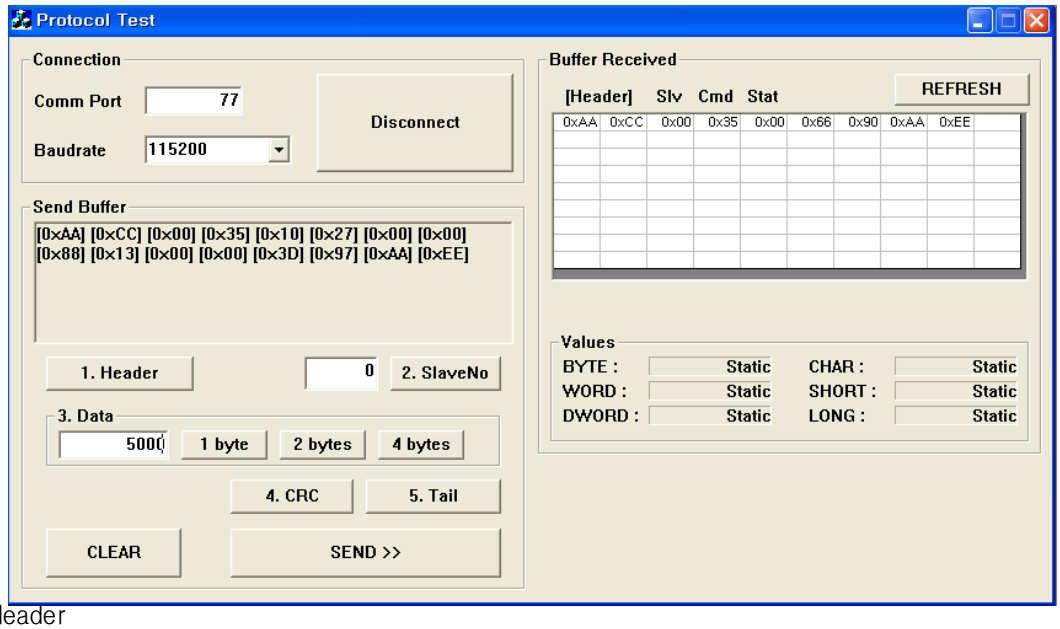
7)Tail : click 'Tail' and you can see '[0xAA][0xEE]' on 'Send Buffer' window.

8)Finally click 'Send' button to send command characters to Ezi-SERVO Plus-R.

You can check the motor torque and LED flash for Servo ON status.

9)After sending command, you can check the answering information from Ezi-SERVO Plus-R on 'Buffer Received' window.

(2)Motion command purpose of command



1) Header
2) Slave No.
3) Sync No.
4) Frame type : insert '53' in 1 byte size for 'Incremental Move' command.
5) Data(Position value) : insert '10000' and click '4byte'.
6) Data(Running speed) : insert '5000' and click '4 byte'.
7) CRC
8) Tail
9) Send : When parameter sets as 'default' value, motor rotates as one revolution. '53' command is incremental move command so once click 'Send', motor will rotate again as same distance.

(3)Protocol Programming

In 'Protocol test GUI' automatically calculate the 'Byte stuffing' and 'CRC' data. For protocol programming in PLC, you have to add the function of 'Byte stuffing' and 'CRC' calculation.

For 'Byte stuffing' refer to '1-1-2. RS-485 Communication Protocol' and for 'CRC' refer to '1-1-3. CRC Calculation Example' on UserManual(EziSERVO PlusR)_Communication Function.

FASTECH

FASTECH Co., Ltd.

Rm #1202, Bucheon Technopark 401 Dong, Yakdae-dong,
Wonmi-Gu, Bucheon-si, Gyeonggi-do, Rep. Of Korea(Zip:420-734)
TEL : 82-32-234-6300, 6301 FAX : 82-32-234-6302
Email : fastech@fastech.co.kr Homepage : www.fastech.co.kr

- Please note that the specifications are subject to change without notice due to product improvements.

ⓒ Copyright 2008 FASTECH Co.,Ltd.
All Rights Reserved. Mar 17, 2010 Rev.08.05.09